UNIVERSITY
OF TRENTO

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.dit.unitn.it

HIGHEST PROBABILITY SVM NEAREST NEIGHBOR CLASSIFIER FOR SPAM FILTERING

Enrico Blanzieri and Anton Bryl

March 2007

Technical Report # DIT-07-007

Highest Probability SVM Nearest Neighbor Classifier for Spam Filtering

Enrico Blanzieri
University of Trento, Italy
blanzier@dit.unitn.it
and
Anton Bryl
University of Trento, Italy;
Create-Net, Italy
abryl@dit.unitn.it

14 March 2007

Abstract

In this paper we evaluate the performance of the highest probability SVM nearest neighbor classifier, which is a combination of the SVM and k-NN classifiers, on a corpus of email messages. To classify a sample the algorithm performs the following actions: for each k in a predefined set $\{k_1, ..., k_N\}$ it trains an SVM model on k nearest labelled samples, and uses this model to classify the given sample, then fits a sigmoid approximation of the probabilistic output for the SVM model, and computes the probabilities of the positive and the negative answers; than it selects that of the $2 \times N$ resulting answers which has the highest probability. The experimental evaluation shows, that this algorithm is able to achieve higher accuracy than the pure SVM classifier at least in the case of equal error costs.

1 Introduction

The problem of unsolicited bulk email, otherwise called *spam*, is today well-known to every user of the Internet. Spam not only causes resource misuse that leads to financial losses, but it is also often used to advertise illegal goods and services or to promote online frauds [14]. According to Siponen and Stucke [17], who presented a study of practical use of anti-spam, the most popular way of anti-spam protection is spam filtering. Many classification algorithms are proposed in order to have accurate and reliable spam filtering, the Support Vector Machine (SVM) classifier being among the best [8, 11, 19].

In this paper we evaluate the performance of a spam filter based on of the highest probability SVM nearest neighbor classifier, which is an improvement over the SVM nearest neighbor classifier. SVM nearest neighbor (SVM-NN) [4] is a combination of the SVM and k-NN classifiers. In order to classify a sample, it first selects k training samples nearest to the given one, and then uses this k samples for training an SVM model which is further used to make the decision. This method is able to achieve a smaller generalization error bound in comparison to pure SVM because of bigger margin and a smaller ball containing the points. The motivation for using this classifier for spam filtering is the following. Spam is known to be not uniform, but rather to consist of messages on different topics [9] and in different genres [7]. The same can be applied also to legitimate mail. This suggests that a classifier which works on a local level can achieve good results on this data. However, this algorithm proposes no way of estimation of the parameter k. Blanzieri and Bryl [3] made an attempt to estimate k by internal training and testing

on the training data. This approach, however, brought uncertain results. Instead, we propose a way to select the parameter k for each sample separately, using the sigmoid approximation to transform the output of SVM into posterior probabilities [15]. The experimental evaluation performed by us shows that the proposed algorithm is able to outperform the pure SVM classifier at least in the case of equal error costs.

The rest of the paper is organized as follows. In Section 2 we give a brief overview of the related work, in Section 3 we present the description of the algorithm, Section 4 contains description and discussion of the experiments, and Section 5 is a conclusion.

2 Related Work

Starting with the Naïve Bayes classifier [16], a great number of classifiers based on the machine learning paradigm were proposed for spam filtering. Some of them exploit the knowledge about the message header structure, retrieving particular kinds of technical information and classifying messages according to it. For example, Leiba et al. [12] propose to analyze IP addresses in the SMTP path contained in the header of the message. Another group of approaches to spam filtering uses human language technologies to analyze the content of the message, for example the approach proposed by Medlock [13] is based on smooth n-gram language modelling.

Still, there is a large group of learning-based spam filters, that observe an email message just as a set of tokens. Such filters can use data from the message content, or from the message header, or from both. The Naïve Bayes filter belongs to this group. Also, the Support Vector Machine (SVM) classifier was proposed for spam filtering by Drucker et al. [8], and proved to show good results in comparison to other methods [8, 11, 19]. We must also mention here the filtering approaches based on maximum entropy model [18] and boosting [8], both showing accuracy comparable to that of SVM [19]. A spam filter based on the k-nearest neighbor (k-NN) algorithm was introduced by Androutsopoulos et al. [1]. The k-NN classifier showed quite poor results on the spam filtering task [11, 19]. For a more detailed overview of the existing approaches to anti-spam protection see the survey by Blanzieri and Bryl [2].

3 The Algorithm

In order to present the highest probability SVM nearest neighbor classifier we need first to present briefly the SVM classifier and the SVM nearest neighbor classifier.

3.1 Support Vector Machines

Support Vector Machine (SVM) is a state of the art classifier [5]. Below we give a brief description of it. For a more detailed description see, for example, the book by Cristianini and Shawe-Taylor [6]. Let there be n labelled training samples that belong to two classes. Each sample x_i is a vector of dimensionality d, and each label y_i is either 1 or -1 depending on the class of the sample. Thus, the training data set can be described as follows:

$$T = ((x_1, y_1), (x_2, y_2), ..., (x_n, y_n)),$$

$$x_i \in \mathbb{R}^d, y_i \in \{-1, 1\}.$$

Given this training samples and a predefined transformation $\Phi : \mathbb{R}^d \rightarrow F$, which maps the features in a transformed feature space, the classifier builds a decision rule of the following form:

$$y_p(x) = sign\left(\sum_{i=1}^{L} \alpha_i y_i K(x_i, x) + b\right),$$

where $K(a, b) = \Phi(a) \cdot \Phi(b)$ is the kernel function and α_i and b are selected so as to maximize the margin of the separating hyperplane. It is also possible to get the result not in the binary form, but in the form of a real number, by dropping the sign function:

$$y'_p(x) = \sum_{i=1}^{L} \alpha_i y_i K(x_i, x) + b,$$

Such real-number output can be useful for adjusting the balance between the two types of errors [19] and, as it will be discussed further, for estimating the posterior probability of the classification error.

In some classification applications it is required that the output of the classifier is not a binary decision, but a posterior probability $P(class|input)$. SVM provides no direct way to obtain such probabilities. Anyhow, several ways of approximation of the posterior probabilities for

Algorithm: The SVM Nearest Neighbor Classifier

Require: sample x to classify;
training set $T = \{(x_1, y_1), (x_2, y_2), ...(x_n, y_n)\}$;
number of nearest neighbors k.

Ensure: decision $y_p \in \{-1, 1\}$

1: Find k samples (x_i, y_i) with minimal values of $K(x_i, x_i) - 2 * K(x_i, x)$
2: Train an SVM model on the k selected samples
3: Classify x using this model, get the result y_p
4: return y_p

Figure 1: The SVM Nearest Neighbor Classifier: pseudocode.

SVM are proposed in the literature. In particular, Platt [15] proposed to approximate the posterior probability of the $\{y = 1\}$ class with a sigmoid:

$$P(y = 1|y'_p) = \frac{1}{1 + e^{Ay'_p + B}} \quad (1)$$

where A and B are the parameters obtained by fitting on an additional training set. Platt observes, that for the SVM classifier with the linear kernel it is acceptable to use the same training set both for training the SVM model and for fitting the sigmoid. The description of the procedure of fitting the parameters A and B can be found in the Platt's original publication [15].

3.2 The SVM Nearest Neighbor Classifier

The SVM Nearest Neighbor (SVM-NN) classifier [4] is a combination of the SVM and k-NN classifiers. In order to classify a sample x, the classifier first selects k samples nearest to the sample x, and then uses this k samples to train an SVM model and perform the classification. The pseudocode of the basic version of the algorithm is given in Figure 1. Samples with minimal values of $K(x_i, x_i) - 2K(x_i, x)$ are the closest samples to the sample x in the transformed feature space, as can be seen from the following equality:

$$||\Phi(x_i) - \Phi(x)||^2 = \Phi^2(x_i) + \Phi^2(x) - 2\Phi(x_i) \cdot \Phi(x) =$$

$$= K(x_i, x_i) + K(x, x) - 2K(x_i, x)$$

A practical problem with this algorithm is that, as such, it provides no procedure to find a good value for the parameter k. In other works the parameter k was determined by means of a validation set [3, 4]. The approach proved to be useful for remote sensing data where the procedure outperformed SVM but not for spam classification where the results were not conclusive. In both cases the SVM-NN methods outperformed the usual k-NN based on the majority vote.

3.3 The Highest Probability SVM Nearest Neighbor Classifier

The highest probability SVM Nearest Neighbor (HP-SVM-NN) classifier is based on the idea of selecting the parameter k from a predefined set $\{k_1, ..., k_N\}$ separately for each sample x which must be classified. To do this, the classifier first performs the following actions for each considered k: k training samples nearest to the sample x are selected; an SVM model is trained on this k samples; then, the same k samples are classified using this model, and the output is used to fit the parameters A and B in the equation (1); then, the sample x is classified using this model, and the real-number output of the model is used to calculate the estimation of the probabilities $P(non - spam|x)$ and $P(spam|x) = 1 - P(non - spam|x)$; in this way, $2 \times N$ answers are obtained. Then, the answer with the highest posterior probability estimate is chosen. An additional parameter C can be used to adjust the balance between the two types of errors. In this case, the probability of error for the negative answer must be not just lower then the probability of error for the positive answer, but at least C times lower to be selected. Surely, $C < 1$ can be used if false positives are less desirable than false negatives.

It must be mentioned, that from the point of view of

Algorithm: The Highest Probability SVM Nearest Neighbor Classifier

Require: sample x to classify;
training set $T = \{(x_1, y_1), (x_2, y_2), ...(x_n, y_n)\}$;
set of possible values for the number of nearest neighbors $\{k_1, k_2, ..., k_N\}$;
parameter C which allows to adjust the balance between the two types of errors.

Ensure: decision $y_p \in \{-1, 1\}$

Order the training samples by the value of $K(x_i, x_i) - 2 * K(x_i, x)$ in ascending order.
MinErr = 1000
Value = 0
if the first k_1 values are all from the same class c **then**
  return c
end if
for all k **do**
  Train SVM model on the first k training samples in the ordered list.
  Classify x using the SVM model, get the result y'_p.
  Classify the same training samples using this model.
  Fit the parameters A and B for the estimation of the P(y=1—y'_p).
  ErrorPositive = (1 - P(y=1—y'_p))
  ErrorNegative = P(y=1—y'_p)
  if ErrorPositive $<$ MinErr **then**
    MinErr = ErrorPositive
    Value = 1
  end if
  if ErrorNegative * $C <$ MinErr **then**
    MinErr = ErrorNegative * C
    Value = -1
  end if
end for
return Value

Figure 2: The Highest Probability SVM Nearest Neighbor Classifier: pseudocode.

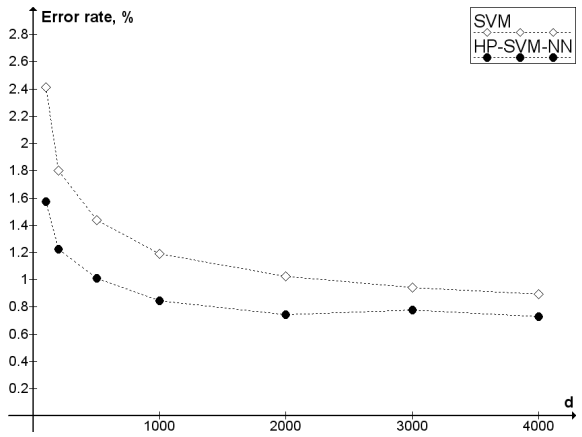

Figure 3: Comparison of SVM and HP-SVM-NN in terms of error rate. *SVM* is the pure SVM classifier, as implemented in SVM*light*. *HP-SVM-NN* is the highest probability SVM-NN classifier, as described in section 3.3. d is the number of features. $C = 1$.

speed such algorithm is not the same as N runs of basic SVM-NN classifier, because the costly operation of distance calculation is performed only once for each classified sample. Nevertheless, in this form the algorithm is very slow and needs some optimization to allow practical usage or fast experimental evaluation. The possibility for such optimization comes from the fact that with the smallest considered k for some samples all the nearest neighbors selected are from the same class. If such case occurs, it is considered sufficient evidence for taking the decision immediately and performing no future search. This version of the algorithm performs faster then the initial one, though obviously much slower that the pure SVM classifier nevertheless. The pseudocode of this optimized version of the algorithm is presented on the Figure 2.

4 Experimental Evaluation

4.1 Experimental Procedures

In order to evaluate the performance of the proposed algorithm and to compare it to the pure SVM classifier, we established two experiments, both of them using ten-fold cross-validation on the SpamAssassin corpus[1]. The corpus contains 4150 legitimate messages and 1897 spam messages (spam rate is 31.37%). The partitioning of data is the same for all the runs. The linear kernel was used in both classifiers. The set of 25 possible values of the parameter k ranges from 34 to 5334 with increasing steps. The following values of the number of features d were used: 100, 200, 500, 1000, 2000, 3000, 4000.

Feature extraction for SVM-based classification of email can be performed in different ways. In the experiment described below we used the following procedure. Each part of the message, namely the message body and each field of the message header, is considered as an unordered set of strings (tokens) separated by spaces. Presence of a certain token in a certain part of a message is considered a binary feature of this message. Then, the d features with the highest information gain are selected. The information gain of a feature is defined as follows:

$$IG(f_k) =$$

$$\sum_{c \in \{c_1, c_2\}} \left(\sum_{f \in \{f_k, \neg f_k\}} P(f, c) \times log \frac{P(f, c)}{P(f) \times P(c)} \right),$$

where f is a binary feature, and c_1 and c_2 are the two classes of samples. Thus, each message is represented with a vector of d binary features.

The implementation of SVM used for this experiment is a popular set of open-source utilities called SVM*light*[2] [10]. Feature extraction is performed by a Perl script. The HP-SVM-NN classifier is implemented as a Perl script which uses SVM*light* utilities as external modules for SVM training and classification.

In the first experiment we compare the performance of the classifiers assuming that the error costs are equal. The parameter C is thus chosen equal to 1. The results of the comparison in terms of overall error rate are presented in Figure 3. The same results in terms of false positives (non-spam classified as spam) and false negatives (spam classified as non-spam) are given in Figure 4.

In the second experiment we compare the performance of the classifiers assuming that false positives are 9 times more expensive than false negatives. To adjust the relative

[1] Available at: http://spamassassin.apache.org/publiccorpus/

[2] Available at: http://svmlight.joachims.org/

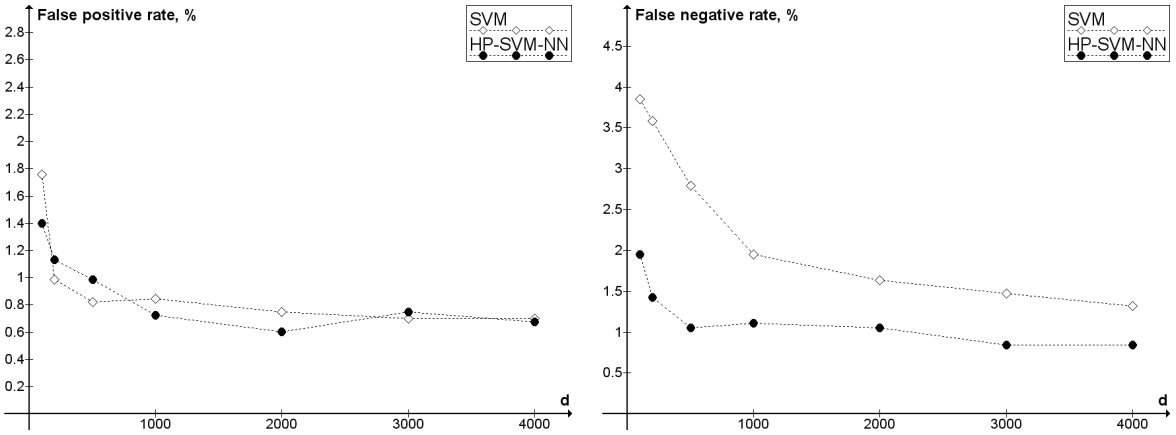

Figure 4: Comparison of SVM and HP-SVM-NN in terms of false positive rate and false negative rate. *SVM* is the pure SVM classifier, as implemented in SVM*light*. *HP-SVM-NN* is the highest probability SVM-NN classifier, as described in section 3.3. d is the number of features. $C = 1$.

cost of errors for the HP-SVM-NN classifier, we select the parameter $C = \frac{1}{9}$. For the SVM classifier we use the possibility of adjusting the relative error costs given by SVM*light*. Weighted error rate is used instead of simple error rate in this case to judge the performance:

$$WErr_9 = \frac{9 \times n_{fp} + n_{fn}}{n_p + 9 \times n_n}$$

where n_{fp} is the number of false positives, n_{fn} is the number of false negatives, n_p is the total number of positives (spam), and n_n is the total number of negatives (non-spam). The results in terms of weighted error rate are given in the Figure 5. The same results in terms of false positives and false negatives are given in Figure 6.

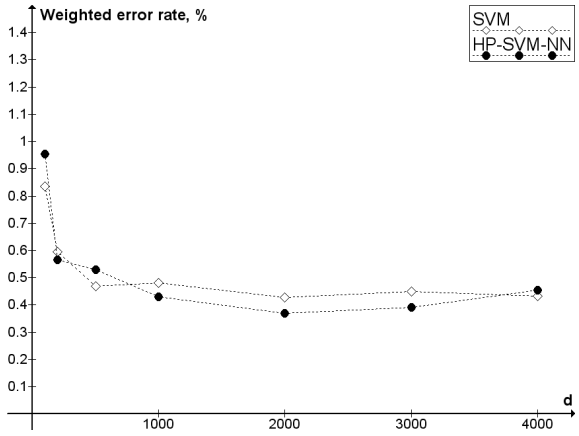

Figure 5: Comparison of SVM and HP-SVM-NN in terms of weighted error rate. *SVM* is the pure SVM classifier, as implemented in SVM*light*. *HP-SVM-NN* is the highest probability SVM-NN classifier, as described in section 3.3. d is the number of features. False positives are considered 9 times more expensive than false negatives. $C = 1/9$.

4.2 Results

As we can see in Figure 3, the highest probability SVM nearest neighbor classifier achieves higher accuracy with all the considered values of d. The evaluation of significance of this advantage, performed using Wilcoxon signed-rank test with $\alpha = 0.05$, showed that significance is present for all used values of d lower or equal to 2000, and not present for $d = 3000$ and $d = 4000$. The comparison in terms of two types of errors (4) shows that the

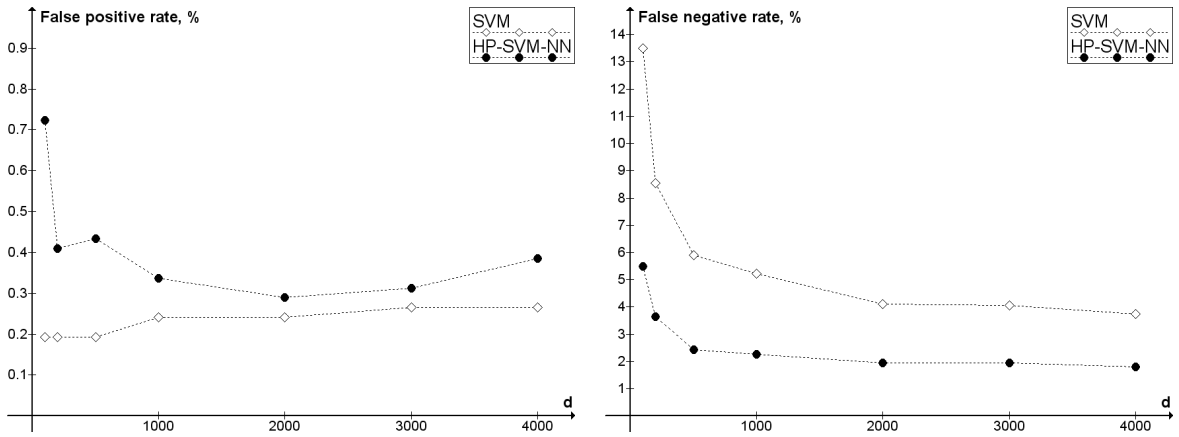

Figure 6: Comparison of SVM and HP-SVM-NN in terms of false positive rate and false negative rate. *SVM* is the pure SVM classifier, as implemented in SVM*light*. *HP-SVM-NN* is the highest probability SVM-NN classifier, as described in section 3.3. d is the number of features. False positives are considered 9 times more expensive than false negatives. $C = 1/9$.

classifiers have close performance in terms of false positives, but the HP-SVM-NN classifier is invariably better in term of false negatives.

However, in the case of unequal error costs, as it can be seen of Figure 5, none of the classifier achieves clear advantage. The result is slightly better for HP-SVM-NN with $d = 1000$, $d = 2000$ and $d = 3000$, but without being significantly different by means of Wilcoxon test.

Being more accurate that SVM, our classifier is at the same time obviously much slower. Our implementation, consisting of a Perl script and external modules that implement SVM, with the number of features $d = 500$ classifier an optimized "easy" sample in about a second and a usual sample in about twenty seconds on a PC with CPU speed of 2.50GHz. It must be mentioned, however, that there is much space for optimization at the software level, by which serious, though not crucial, improvement on speed can be achieved. Also, it is possible to use a smaller set of possible values of k.

5 Conclusion

In this paper we proposed and discussed the highest probability SVM nearest neighbor classifier. The classifier is a local SVM classifier, based on k samples in the neighborhood of the sample to classify, k is selected dynamically among a pool of values depending on an estimate of the a-posteriori probability done following Platt [15]. Experimental comparison with the pure SVM classifier is performed, showing that our classifier is able to achieve better accuracy than SVM in the case of equal error costs. Thus locality proved to be a viable way of increasing the accuracy of the classification at the price of extra computation. In the experiment with unequal error cost, however, no significant difference is achieved, and further experimental evaluation and improvement of the procedure of selecting the parameter k is needed.

References

[1] I. Androutsopoulos, G. Paliouras, V. Karkaletsis, G. Sakkis, C. Spyropoulos, and P. Stamatopoulos. Learning to filter spam e-mail: A comparison of a naive bayesian and a memory-based approach. In H. Zaragoza, P. Gallinari, and M. Rajman, editors, *Proceedings of the Workshop on Machine Learning and Textual Information Access, 4th European Conference on Principles and Practice of Knowledge Discovery in Databases, PKDD 2000*, pages 1–13, 2000.

[2] E. Blanzieri and A. Bryl. A survey of anti-spam techniques. Technical report #DIT-06-056. 2006.

[3] E. Blanzieri and A. Bryl. Instance-based spam filtering using SVM nearest neighbor classifier. In *Proceedings of FLAIRS 2007 (to be published)*, 2007.

[4] E. Blanzieri and F. Melgani. An adaptive SVM nearest neighbor classifier for remotely sensed imagery. In *Proceedings of 2006 IEEE International Geoscience And Remote Sensing Symposium*, 2006.

[5] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[6] N. Cristianini and J. Shawe-Taylor. *Support Vector Machines*. Cambridge University Press, 2000.

[7] W. Cukier, S. Cody, and E. Nesselroth. Genres of spam: Expectations and deceptions. In *Proceedings of the 39th Annual Hawaii International Conference on System Sciences, HICSS '06*, volume 3, 2006.

[8] H. Drucker, D. Wu, and V. Vapnik. Support vector machines for spam categorization. *IEEE Transactions on Neural networks*, 10(5):1048–1054, 1999.

[9] G. Hulten, A. Penta, G. Seshadrinathan, and M. Mishra. Trends in spam products and methods. In *Proceedings of the First Conference on Email and Anti-Spam, CEAS'2004*, 2004.

[10] T. Joachims. *Making large-Scale SVM Learning Practical*. MIT-Press, 1999.

[11] C.-C. Lai and M.-C. Tsai. An empirical performance comparison of machine learning methods for spam e-mail categorization. *Hybrid Intelligent Systems*, pages 44–48, 2004.

[12] B. Leiba, J. Ossher, V. T. Rajan, R. Segal, and M. Wegman. SMTP path analysis. In *Proceedings of Second Conference on Email and Anti-Spam, CEAS'2005*, 2005.

[13] B. Medlock. An adaptive approach to spam filtering on a new corpus. In *Proceedings of the Third Conference on Email and Anti-Spam, CEAS'2006*, 2006.

[14] E. Moustakas, C. Ranganathan, and P. Duquenoy. Combating spam through legislation: A comparative analysis of us and european approaches. In *Proceedings of Second Conference on Email and Anti-Spam, CEAS'2005*, 2005.

[15] J. Platt. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In A. Smola, P. Bartlett, B. Schoelkopf, and D. Schuurmans, editors, *Advances in Large Margin Classiers*, pages 61–74, 2000.

[16] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*. AAAI Technical Report WS-98-05, 1998.

[17] M. Siponen and C. Stucke. Effective anti-spam strategies in companies: An international study. In *Proceedings of the 39th Annual Hawaii International Conference on System Sciences, HICSS '06*, volume 6, 2006.

[18] L. Zhang and T. Yao. Filtering junk mail with a maximum entropy model. In *Proceeding of 20th International Conference on Computer Processing of Oriental Languages, ICCPOL03*, pages 446–453, 2003.

[19] L. Zhang, J. Zhu, and T. Yao. An evaluation of statistical spam filtering techniques. *ACM Transactions on Asian Language Information Processing (TALIP)*, 3(4):243–269, 2004.